Attention Network for 3D Object Detection in Point Clouds

Anshul Paigwar

Master of Science in Informatics at Grenoble
Master Mathématiques Informatique - spécialité Informatique
option Graphics, Vision and Robotics (GVR)

Attentional PointNet for 3D Object Detection in Point Clouds

Anshul Paigwar

3 September, 2018

The research project performed at Chroma-Inria, Grenoble

Under the supervision of:

Prof. Christian Wolf

Prof. Christian Laugier

Ozgur Erkent (Mentor)

Defended before a jury composed of:

Prof. James Crowley

Prof. Dominique Vaufreydaz

Amaury Negre

Abstract

Accurate detection of objects in 3D point clouds is a central problem for autonomous navigation. Most existing methods use techniques of hand-crafted features representation or multi-modal approaches prone to sensor failure. Approaches like PointNet that directly operate on sparse point data have shown good accuracy in the classification of single 3D objects. However, LiDAR sensors on Autonomous vehicles generate a large scale pointcloud. Real-time object detection in such a cluttered environment still remains a challenge. In this thesis, we propose Attentional PointNet, a novel end-to-end trainable deep architecture for object detection in point clouds. We extend the theory of visual attention mechanism to 3D point clouds and introduce a new recurrent 3D Spatial Transformer Network module. Rather than processing whole point cloud, the network learns "where to look" (find regions of interest), thus significantly reducing the number of points and hence, inference time. Evaluation on KITTI car detection benchmark shows that our Attentional PointNet is notably faster and achieves comparable results with *state-of-the-art* LiDAR-based 3D detection methods.

Résumé

La détection précise d'objets dans un nuage de points 3D est un problème central pour la navigation autonome. La plupart des méthodes existantes utilisent des caractéristiques sélectionnées à la main ou des approches multi-modèles sujettes à une défaillance du capteur. Des approches, telles que PointNet fonctionnant directement sur des données ponctuelles éparses, classifient précisément un nuage de points associé à un unique objet. Cependant, les capteurs Lidars sur les véhicules autonomes génèrent un nuage de points contenant de nombreux objets. Leurs détections en temps réel dans un environnement aussi encombré restent un défi. Dans cette thèse, nous proposons une méthode appelée Attentional PointNet, une architecture profonde complète, formable de bout en bout, destinée à la détection d'objets dans le nuage de points. Nous étendons la théorie du mécanisme d'attention visuelle au

nuage de points 3D et introduisons un nouveau module récurrent de réseau de transformateur spatial 3D. Plutôt que de traiter le nuage de points dans sont ensemble, il apprend à reconnaître des régions potentiellement intéressantes. Ensuite, localiser des objets dans ces régions réduit considérablement le nombre de points à traiter et réduit le temps de calcul. L'évaluation avec les données du jeu de données KITTI montre que notre méthode est plus rapide et permet d'obtenir des résultats comparables avec les méthodes classiques de détection 3D utilisant des nuages de points générés par des Lidars

Acknowledgement

I would like to take this opportunity to express my sincere gratitude and appreciation to everyone who supported me during this program.

First of all, I would like to thank my supervisor Prof. Christian Laugier for his constant support and motivating me to pursue higher education in Master of Science in Robotics program.

I would also like to thank my second supervisor, Prof. Christian Wolf, for feeding my curiosity with state-of-the-art deep learning techniques, supporting me technically and steering me in the right the direction whenever I needed it.

I would also like to thank my mentor Ozgur Erkent for constantly pushing me to do better and always being open for discussion and brainstorming over my silly questions and ideas.

I express my sincere gratitude to all the member of Team CHROMA, Inria, especially David Sierra Gonzalez, Pavan Vasishta and Mathieu Barbier for their continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

Finally, I must express my very profound gratitude to my parents and to my friends for providing me with unfailing support. This accomplishment would not have been possible without them. Thank you.

Contents

1 Introduction

> *The last thing that we find in making a book is to know what we must put first.*
>
> **— Blaise Pascal**
> (Mathematician, physicist, inventor)

From high-speed autonomous vehicles that navigate on busy crossroads [1], to mobile robots that sweep the floor in your home [2], to humanoid robots that serve you food at a restaurant, or quad-copters mapping and inspecting a industrial factory rely on three-dimensional (3D) data of physical surrounding. With a rapid upsurge in development of lasers, highly compact and yet affordable laser scanners (LiDARs) are now available. These LiDARs generate high quality and reliable 3D data in the form of point clouds.

Another popular approach of generating point clouds is from stereo cameras. Most mobile phone now comes equipped with dual camera setup, allowing them to generate point clouds for applications like Augmented Reality and Virtual Reality [3]. Point clouds generated from cameras are comparatively inferior to LiDARs [4].

Consider the application of humanoid robot, serving food in a restaurant, which would require to first map the environment and localize itself. Then it needs to detect various external obstacles like humans, tables and chairs to plan its trajectory to avoid them. Many such real-world applications require the localization of obstacles and characterization of their shape.

1.1 Motivation and Problem Statement

Navigation of Autonomous Vehicles is one such principle application where high definition LiDARs are extensively used. One of the most common applications is to generate 2D occupancy grids using LiDAR data as in [5]. Occupancy Grids are then used by path planning algorithms to generate trajectories for navigation. Occupancy grids are essential to finding navigable and non-navigable spaces but each grid cell only contains sparse information

e.g. occupied or non-occupied. A method developed by Inria-Chroma, CM-CDOT [6] also encodes other information like dynamic, static or unknown. However, there is no semantic information, creating no distinction between obstacles (e.g. pedestrian vs lamp post, both are classified as obstacles). The planner generates the trajectories considering all types of objects with equal weightage. From the theory of situational modelling [7], driving behaviour near pedestrians differs from the lamp post, and that is the reason why semantic information is an essential part of planning.

Incorporating semantic information into occupancy grids, multiple object detection in 3D data and categorizing them into distinct objects is crucial to this task. Ozgur *et al.* in [8] have used an approach of projecting semantically segmented images on corresponding occupancy grid of the same scene and fusing both using deep learning methods. However, the need for an additional camera that is time synchronized and calibrated with the LiDAR restricts their use and makes the solution more sensitive to sensor failure modes.

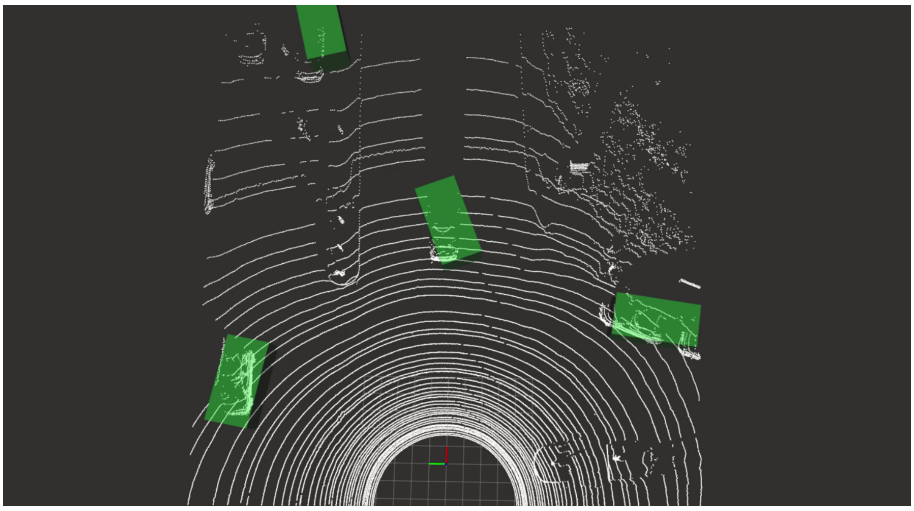


Fig. 1.1: Attentional PointNet: Multiple object detection in 3D point clouds

The research work in this thesis therefore seeks to address the following problems:

- To detect and classify objects in 3D point clouds, obtained by high-definition LiDARs in autonomous vehicles. This also aims at integrating these detected objects into occupancy grid representation.

- To improve computational efficiency to enable real time perception for high-speed autonomous vehicles.

- To avoid the limitations of multi-modal approaches such as sensor failure and affection to changes in environmental condition, the approach must rely only on LiDAR data.

1.2 Deep Learning on Point Clouds

With the rapid development of 3D sensor technology, LiDARs are fastly becoming a key sensor in many robotic applications. Also, the availability of many open sourced, high quality annotated 3D point cloud data has motivated researchers to develop efficient feature representations to detect and localize objects in point clouds[9]. When point cloud data has rich geometric information of the objects it is representing, the hand-crafted features yield satisfactory results. However, their inability to generalize and adapt to more complex shapes and unstructured environment resulted in limited success for autonomous navigation.

To remove manual feature engineering for representation of point clouds, many researchers are using Deep Learning (DL) leveraging its astonishing success in multiple object detection and semantic segmentation in 2D images.

Point clouds represent the outer surface of objects in the scene. In contrast to images where detailed texture information is available, point clouds are sparse, unordered and have highly variable point density. To deal with such challenges, prior work have been based on the following approaches:

- Converting point clouds into 2D images [10, 11] and reinstating the state-of-art deep architectures to detect multiple objects and then projecting results back to 3D space. However, converting point clouds to 2D images results in losing essential 3D structural information of the objects.

- Converting point clouds into volumetric forms like voxel grids [12, 13] and generalizing image CNNs to 3D CNNs. However, for dense 3D data, computational and memory requirements grow cubically with the resolution of voxels. While Riegler *et al* in [14, 15] have used octrees representation to exploit the sparsity of point clouds, these CNN based methods still require quantization of point clouds with certain voxel resolution.

- Another approach is inferring 3D bounding boxes directly from 2D images [16]. However, the depth estimation greatly affects the accuracy of image-based 3D detection.

Several other studies involve multi-modal fusion [17, 11, 8] that combine images and LiDAR data to improve detection accuracy particularly for small objects (pedestrians, cyclists).

Recent work [18, 19, 20, 21, 22] proposes novel types of network architectures which directly consumes raw point clouds without converting them to other formats. Among these PointNet [18], being simpler and real-time, have shown encouraging results for single object classification and semantic segmentation. In this thesis, we have also studied another 3D classifier PointWise Convolution [20], and compared its performance with PointNet.

Some very recent work [23, 24] explore how to extend the PointNet architecture for the purpose of multiple 3D object detection in large-scale 3D point cloud data obtained from high definition LiDARs. Charles *et al.* in frustum PointNet [23] have used a multi-modal approach of first detecting objects in 2D images, reducing the search space and then regressing corresponding 3D bounding boxes using variants of PointNet.

Currently, VoxelNet [24] from researchers at Apple, is the only end-to-end trainable deep network which takes LiDAR point clouds directly and outputs 3D bounding box predictions for multiple objects in uncontrolled environments. VoxelNet outperforms the state-of-the-art LiDAR-based 3D detection methods by a large margin. However, VoxelNet architecture is complex and requires higher compute capability hardware for real-time implementation.

This thesis focuses on exploring alternate methods for LiDAR only detection. We aim to design simpler architecture with real-time performance on lower compute capability hardware.

1.3 Contributions

- We propose a novel deep architecture called Attentional PointNet for 3D object detection. The network directly operates on sparse 3D points and is end-to-end trainable.

- We extend the theory of Visual Attention Mechanism to 3-dimensional space for multiple object detection. Given a cluttered environment, we show that the network learns to attend to objects of interest thus reducing the data needed to be processed.

- We experiment with various Spatial Transformer Network(STN) based differentiable Visual Attention Mechanisms on the cluttered MNIST dataset and evaluate their performance.

- We experiment with different 3D object Classifiers and evaluate their performance. We also elaborate on how to efficiently Implement the PointWise Convolution Network in PyTorch Framework.

- We conduct experiments on KITTI benchmark and show that Attentional PointNet achieves real-time performance and comparable results in LiDAR-based car detection methods.

1.4 Thesis Structure

Chapter 2

This chapter provides more general introduction to point clouds, their properties, feature representation and associated difficulties in context with Deep Learning. Then we describe various approaches employed for multiple object detection in 2D Images. We elaborate on one approach of Visual Attention mechanism and its differentiable variants using spatial Transformer Network. We comment on how using Attention mechanism in 3D space could save computational efforts. Finally, we study the architectures of PointNet and PointWise Convolution networks and discuss on their performance.

Chapter 4

This chapter briefs about the datasets used for the experimentation in this thesis. Namely, Cluttered MNIST dataset for Attention Mechanisms in images, ModelNet40, KITTI 3D dataset and ACFR urban dataset for comparison between PointNet and PointWise Convolution Network. Finally, we explain in detail the strategy used generating dataset to train proposed Attentional PointNet architecture.

Chapter 3

In this chapter we explain the idea behind the proposed architecture of Attentional PointNet. We then break down the architecture into smaller functioning blocks and explain each block in detail. We also supplement each block with mathematical analysis. Finally we discuss about the proposed loss function to the train the network.

Chapter 5

This chapter describes three experiments, first for finding the best differentiable Visual Attention Network. Second, the most suitable 3D Classifier. Third, experimentation with proposed Attentional PointNet. We mention corresponding values of hyper parameters chosen for each network. Finally we evaluate our networks on standard parameters and provide comparison results with other state of the art methods.

Chapter 6

In this chapter we argue about the contributions of this work, its scope and give conclusions. We recommend ideas for future work and changes that might lead to improved accuracy.

2 Related Work and Theoretical Background

> *Know how to solve every problem that has ever been solved.*
>
> — **Richard Feynman**
> (Quantum physicist)

Object detection is a central task in computer vision, with applications ranging across search, robotics, self-driving cars, and many others. It is a long-standing field of research with researchers trying to formulate more comprehensive feature representations of images and 3D data structures. But most progress has been done only in recent years with the emergence of DL on imagery and its success owed to Convolutional Neural Networks (CNN).

Images are often stored as two-dimensional matrices of pixel values. Convolution network use convolutions to abstract spatial features from the neighbouring pixels. To apply the same principle to point clouds is non-trivial. Point clouds are generally stored as an array of point records and are in no particular order.

In section 2.1 we detail about the properties of point clouds and related issues with using conventional deep learning techniques for object classification. section 2.2 presents new architectures as 3D object classifiers designed to work with point set. section 2.3 briefs about Region Proposal networks and the current state-of-the-art methodology for multiple object detection in images. section 2.4 presents the methodology of Visual attention Mechanism and Spatial Transformer Network based variants. Finally, we try to convince why attention is better and faster than region proposals for object detection in large-scale point clouds.

2.1 Properties of PointCloud

A point cloud is a set of data points in a 3D Euclidean space. In practice, the term is used to refer to 3D points representing the external surface, geometry and location of real-world objects. Every point in a point cloud is represented by it's X, Y and Z coordinate with respect to some reference frame. Depending upon the sensor used to generate pointcloud each point could also be associated with other common attributes like RGB for color or intensity of the laser reflection. These attributes define the properties of the scanned object and can be used for the classification.

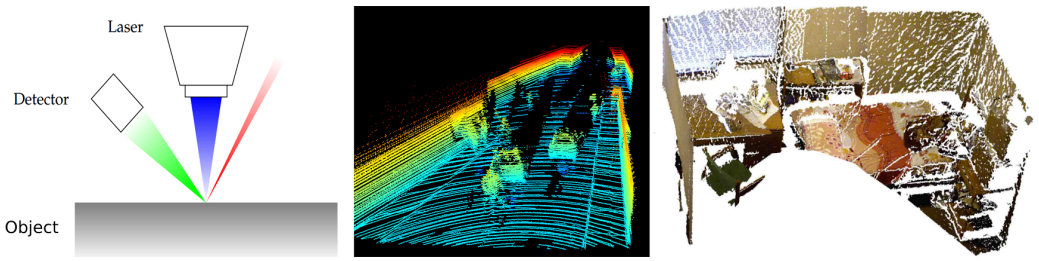


Fig. 2.1: (a) Shows the principle of working of LiDAR; (b) Typical point cloud generated by high-definition LiDAR; (c) Point cloud generated from stereo cameras

Source: (a) TNO Research; (b) IST -TU Graz; (c) Cornell-RGBD-Dataset

Similar to [25] we list challenges associated with point clouds as follows:

2.1.1 Challenges with Neural Networks

- **Unstructured data:** Unlike in images where pixel values are stored in defined 2D matrices, points in pointcloud are stored as a list without any specific order. It means geometrically neighbouring point in 3D space not necessarily be the neighbouring point in the list. This makes the existing CNN filters impotent.

- **Invariance to Permutation:** The order of the points in the list doesn't matter geometrically as it still represents the same shape. However, changing the point order changes underlying matrix structure. So, the same point cloud can be represented by two very different matrices. See Figure 2.2. In other words, a network which takes in N 3D point as input needs to be invariant to N! permutations in order of feeding data.

- **Different number of points:** In images, depending upon the resolution of the camera the number of pixels is constant. Whereas the number of points may vary dramatically, depending on sensor and the changes in the environment being scanned.

- **Varying density of points:** High definition LiDARs on Autonomous Vehicles has an array of laser beams pointed at a certain angle and revolving in 360 degrees as shown in Figure 2.2. Because of this sensor model objects closer to the sensor have higher point count than the same object at the farther distance.

- **Invariance to Transformation:** As a 3D dataset the geometric object could be rotated and translated. The learned representation of object should be invariant to these simple transformations and should not modify its category.

- **Interaction among points:** Even though point set is in an unordered list, these points are in Euclidean space and geometrically neighbouring points form a meaningful subset. The network should encapsulate local structures from nearby points and the combinatorial interactions among them.

- **Missing data and self occlusion:** With the LiDARs because of ray casting sensor model the scanned object would only have the points on the part of the object facing the sensor. If the object is rotating we will have different point clouds that represent the same object see Figure 2.2.

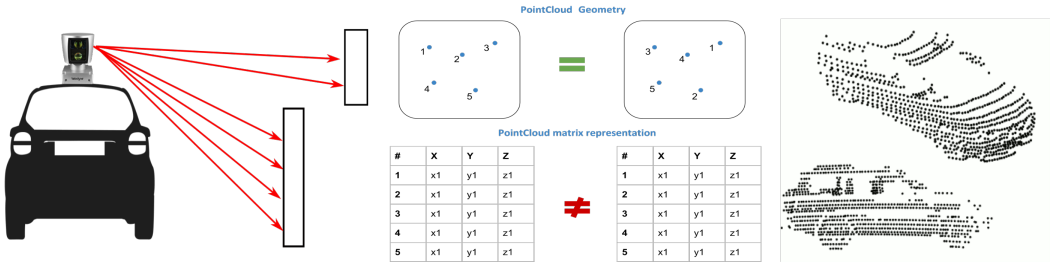


Fig. 2.2: Issues with pointcloud: (a) Varying density of points representing object; (b) Invariance to permutation; (c) Self occlusion with change in orientation

Source: (b) Itzi-blog [25]; (c) ACFR dataset

2.2 3D Classifiers

With increasing popularity and real-world applications of 3D data, there has been a growing interest in the deep learning community for the tasks like object classification and semantic segmentation of point clouds. Many researchers are rising to the challenges associated with the properties of pointcloud and have proposed novel architectures which directly consume points set and learn shape descriptors. In this section we elaborate two such recently proposed architecture PointNet [18] and PointWise Convolution Network [20]. We implemented both the architecture in PyTorch framework and evaluated their performances on different datasets detailed in section 5.1.

2.2.1 PointNet

PointNet [18] proposed by Qi *et al.* is one of the first deep network architecture that can handle point cloud data directly without converting it into other forms of representation. PointNet can take point clouds of arbitrary orders as input and is robust to permutations and transformations. PointNet architecture as shown in Figure 2.3 being simpler have shown impressive results on several tasks such as object classification and semantic segmentation.

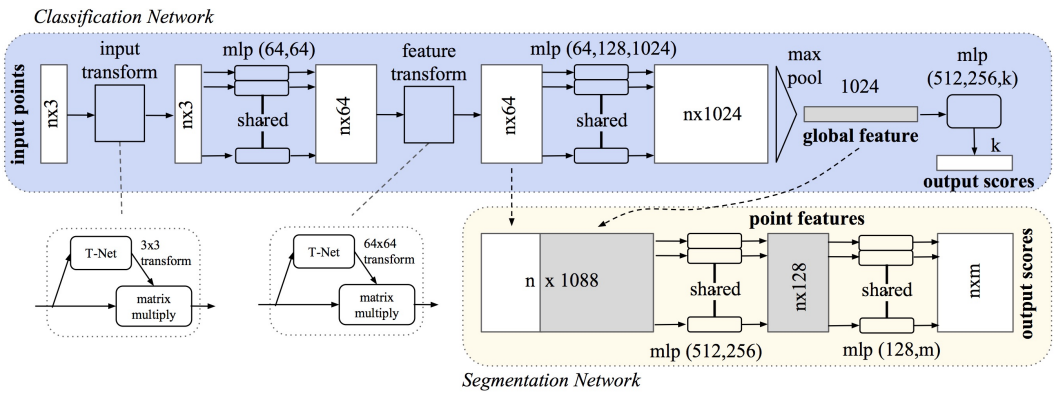


Fig. 2.3: Network architecture of PointNet [18].

PointNet consists of two networks a Joint Alignment Network or T-net and classification Network. The T-net predicts an affine transformation matrix and directly apply this transformation to the coordinates of input points. T-net can be seen as Spatial Transformer Network(STN) [26]for 3D point sets. The T-net itself resembles the big classification network and is a miniature version of it. The T-net is composed of three basic modules of point independent feature extraction, max pooling and fully connected layers.

To be invariant to the permutation of input, PointNet proposes the idea to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set.

$$f(\{x_1, x_2, ..., x_n\}) \approx g(h(x_1), h(x_2), ..., h(x_n)) \tag{2.1}$$

where $f : 2^{\mathbb{R}^N} \rightarrow \mathbb{R}$, $h : \mathbb{R}^N \rightarrow \mathbb{R}^K$ and $g : \mathbb{R}^K \times ... \times \mathbb{R}^K \rightarrow \mathbb{R}$ is a symmetric function.

A series of multi-layered perceptron approximate h by extracting independent point features. While a maxpooling function approximate symmetric function g and aggregate all input point features into one common *global-feature* invariant to permutations. Finally, a fully connected layer regresses parameters of a (3×3) affine Transformation matrix from the global feature.

The classification Network is stacked after T-net. It takes input as the transformed point sets from T-net and is composed of the same basic modules as T-net. The only difference being that in the end the independent point wise features are concatenated with the global feature aggregated by maxpooling layer. Fully connected layers at the end outputs per class probability scores in case of classification task or per point class score for semantic segmentation task.

2.2.2 PointWise Convolution Network

Hua *et al.* in [20] proposes another network architecture that directly works with point sets. PointNet uses a series of multi-layered perceptrons to generate point feature set, it does not exploit completely an important property of point cloud of interaction among points. As point cloud represent geometrical shapes, neighbouring points encodes essential information to understand the 3D structure.

Hua *et al.* focuses on designing new convolution operator, called point-wise convolution, which can be applied at each point in a point cloud to learn point-wise features. The network design as shown in Figure 2.4 is fully convolutional. Multiple pointwise convolution layers can be stacked together to form very deep networks.

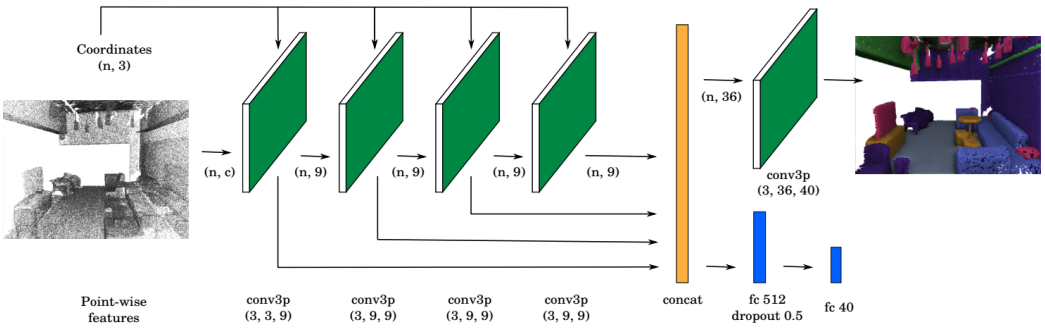


Fig. 2.4: Architecture of Point-wise convolution network [20].

PointWise Convolution Operator: A convolution kernel can be thought as a sphere in 3D space and is centred at each point. All the neighbour points within spherical kernel support contribute to the center point. The spherical kernel is further subdivided into the sub-domains as shown in Figure 2.5. All the points within a sub-domain share same weight values. The goal is to learn these weight values. The radius of the kernel and number of sub-domains are important hyperparameters and can be adjusted to account for the different number of neighbour points in each convolution layer.

$$x_i^l = \sum_k w_k \frac{1}{|\omega_i(k)|} \sum_{p_j \in \omega_i(k)} x_j^{l-1} \tag{2.2}$$

where k iterates over all sub-domains in the kernel support, $\omega_i(k)$ is the k-th sub-domain of the kernel centered at point i; p_i is the coordinate of point i; $|.|$ counts all points within the sub-domain; w_k is the kernel weight at the k-th subdomain, x_i and x_j the value at point i and j, and $l-1$ and l the index of the input and output layer.

However, unlike PointNet which rely on a prefix network to learn a symmetric function to turn a point cloud into a set before performing their further learning and predictions. Point-wise convolution network show that it is possible to perform scene understanding tasks such as semantic segmentation and object recognition on ordered point clouds.

2.3 Multiple Object Detection in images

Multiple object detection is the field where the task is to classify and localize all the object in a scene. Many successful work re-purpose single object

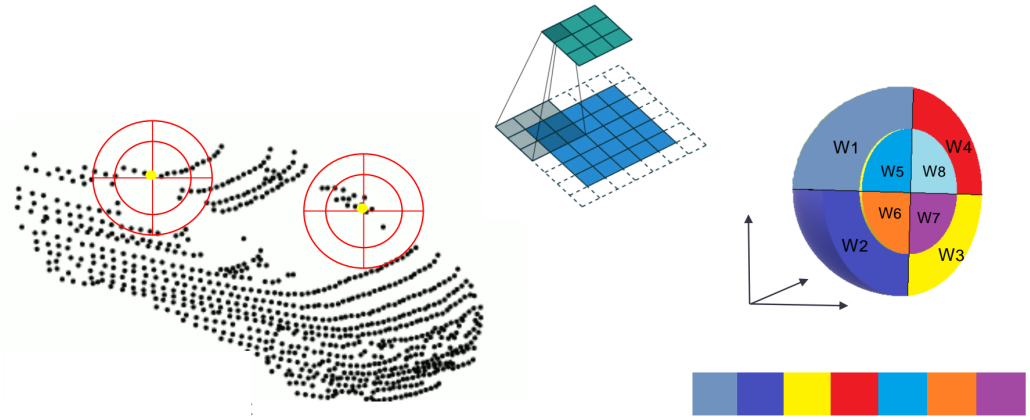


Fig. 2.5: (a)Spherical point-wise convolution operator centred at different points in a point cloud; (b) Convolutions in images; (c) Pointwise kernel and sub-domains; Images reproduced from ACFR dataset and guide to arithmetic convolution

classifiers or localizers to perform multiple detections. For example, in sliding window algorithms a classifier is applied to an image at multiple locations and scales. Regions with high probability scores are considered as detections. Being computationally expensive, yet do not yield accurate detections.

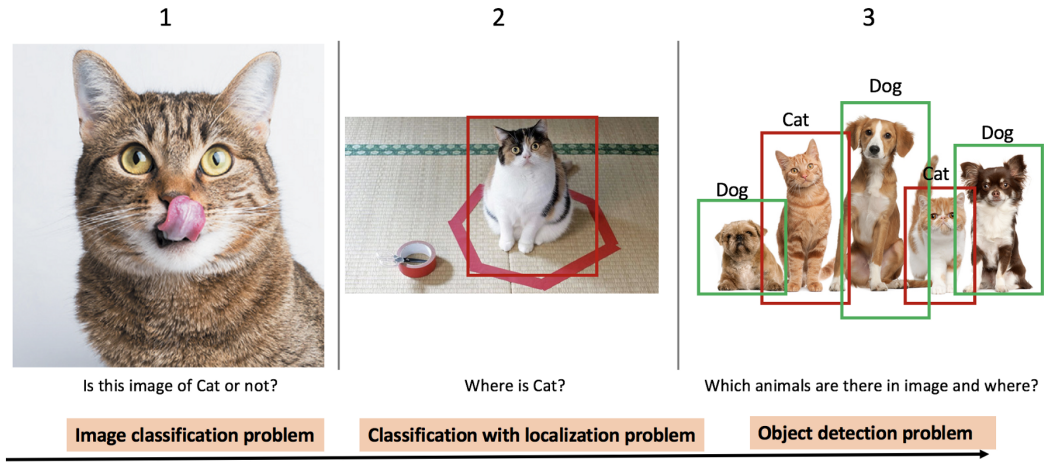


Fig. 2.6: Object Classification, object detection and multiple object detection in images

Source: Google images

Overcoming these problems approaches like Faster R-CNN [27] use region proposals methods where the network first generate potential bounding boxes and then run a classifier only on these proposed boxes. This greatly saves computation as classifiers are not applied to all locations in an image. After classification and post-processing is used to eliminate duplicate detections

and refining bounding boxes. These networks still have complex pipelines and are slow.

More recently other approaches like YOLO [28], SSD [29] have come with approaches which directly predicts bounding boxes and class probabilities in a single network with an end-to-end CNN architecture. While these approaches have shown astonishing results and real-time performance adapting them for detection in large-scale point clouds is not tangible. Difficulties are vastly due to difference in the dataset and properties of point cloud as described in section 2.1.

Above mentioned approaches, all consume and process whole image. In contrast a typical point cloud consist of $\approx 100k$ points in each scan. Most regions, being empty spaces only a few regions are of relevance. Discretizing pointcloud into 3D grid and processing as in whole is futile when aiming for real-time performance. As the computational complexity grows at least linear in the number of pixels or voxels in case of 3D point clouds. In this thesis, we propose to use the theory of visual attention mechanism to overcome these challenges.

2.4 Visual Attention Mechanism

Visual search is extensively involved in everyday perception, and biological systems like the human eye manage to perform it remarkably well. As in [30] human perception does not process the whole scene in its entirety at once. Humans focus to attend relevant parts in the scene acquiring necessary information when and where it is needed. Eventually combining pieces of informations from different fixations and build up representation of the scene [31]. Focusing onto smaller relevant parts of the scene saves "computational bandwidth" as only fewer pixels are needed to be processed. As irrelevant parts in the scene are out of fixation, they are ignored, this reduces the complexity of the task.

Taking Inspiration from human perception of sequentially recognizing objects by moving fovea from one object to the next relevant object, Mnih *et al.* in [32, 33] proposed a deep recurrent neural network that in each iteration processes a multi-resolution crop (glimpse) of input image.

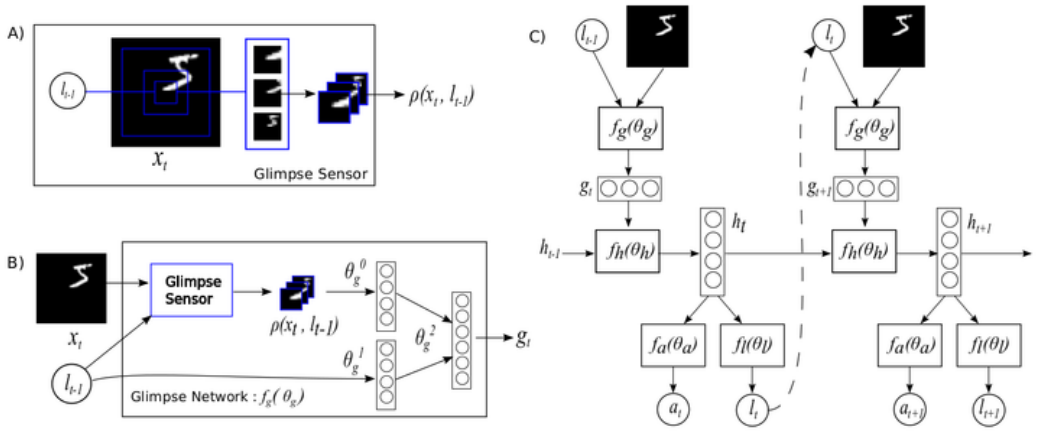


Fig. 2.7: Network architecture of Visual attention mechanism [32].

To learn task-specific strategies like "where to look", the network is modelled as a dynamic visual control problem and a glimpse can be seen as a partial view of the state. Viewing the problem as a POMDP, the techniques from the Reinforcement learning literature can be used here. Neural Network defines a policy which it has to learn. Recurrent part of network plays a vital role as it encapsulates the state history as the hidden state of Network.

This system not being, end to end differentiable, backpropagation is used only to train the neural-network part. While RL based policy gradient is used to address the non-differentiabilities due to the control problem.

2.4.1 Spatial Transformer Network

Selective attention and manipulation of the data by cropping is a non- differentiable operation. Minh *et al* in [32] uses reinforcement learning to avoid the need for a differentiable attention Mechanism. Jaderberg et al in [26] proposed a new Spatial Transformer module which explicitly allows the spatial manipulation of data within the network. Transformations including scaling, cropping, rotations, as well as non-rigid deformations are performed on the entire feature map (non-locally). Spatial transformers can be used to select regions of an image that are most relevant (attention), and transform those regions to expected pose to simplify recognition in subsequent layers.

Spatial Transformer can be seen as diffrentiable spatial attention as they can be trained purely with backpropagation without reinforcement learning. As mentioned earlier attention mechanism benefits from increased computa-

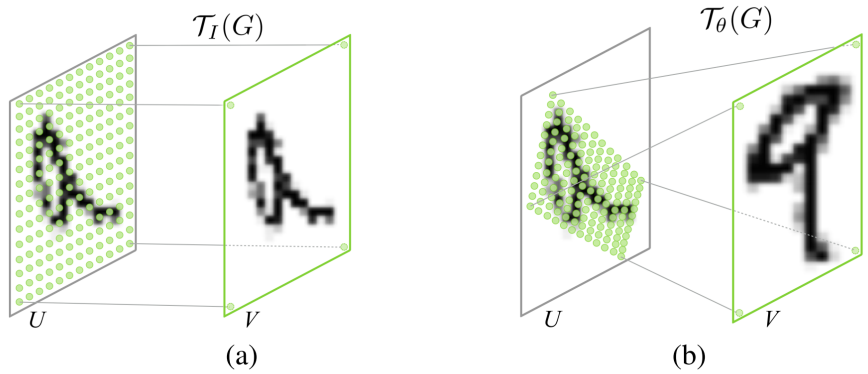


Fig. 2.8: Working of spatial transformer network module and sampling grid [26]

tional efficiency as Transformed (and so attended) lower resolution crops can be used in favour of higher resolution raw inputs

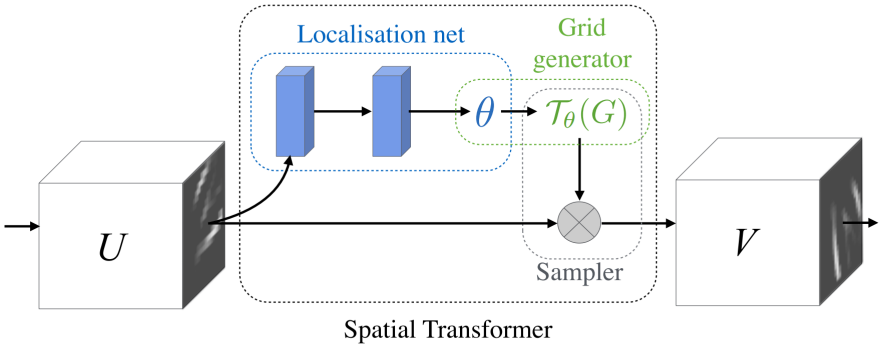


Fig. 2.9: Architecture of Spatial Transformer Network [26]

The spatial transformer module consist of three parts as shown in Figure 2.9.

1. **The localisation network:** It takes input feature map $U \in \mathbb{R}^{H \times W \times C}$ of width W , height H and C channels and regresses θ, the parameters of the transformation $T(\theta)$ to be applied to the feature map:

$$T(\theta) = f_{loc}(U)$$

2. **Parameterised Sampling Grid:** It takes the parameters of the transformation $T(\theta)$ from localisation network and computes a parameterised output grid $V \in \mathbb{R}^{H' \times W' \times C}$ of width W' , height H' and C channels

by applying a sampling kernel centered at a particular location in the input feature map U as shown in Figure 2.8. Let $T\theta$ be the affine transformation A_θ then the pointwise transformation for 2D grid is given by:

$$\begin{bmatrix} x_i^s \\ y_i^s \\ 1 \end{bmatrix} = T_\theta(P_i) = A_\theta \begin{bmatrix} x_i^t \\ y_i^t \\ 1 \end{bmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^t \\ y_i^t \\ 1 \end{bmatrix} \tag{2.3}$$

where (x_i^t, y_i^t, z_i^t) are the target coordinates of the regular grid in output feature map, $(x_i^s; y_i^s, z_i^s)$ are the source coordinates of the input feature map. The transform defined in the Equation 2.3 allows cropping, translation, rotation, scale, and skew to be applied to the input feature map, and requires only 6 parameters to be predicted by localization Network.

3. **Differentiable Image Sampling:** Each $(x_i^s; y_i^s, z_i^s)$ coordinate in $T\theta(G)$ defines the spatial location in the input where a sampling kernel is applied to get the value at a particular pixel in the output V. But pixels may not have one to one correspondence. To solve this problem, a diffrentiable bilinear interpolation technique is used.

2.4.2 Differentiable Visual Attention Mechanism

Building upon the theory of Recurrent attention Mechanism and Spatial Transformer network EDRAM [34] and Recurrent-STN [35] has proposed two differentiable attention mechanism architecture as shown in Figure 2.10. These improved attention-based architectures can localize and recognize multiple object simultaneously and can be trained end to end by backpropagation.

Key differences in both the architectures can be summarized as:

EDRAM architecture resembles [33] where cropping/attention module is at the beginning and rest of the network only sees the cropped portion of the original image. Based on available information network predicts new location to attend.

Recurrent-STN network at first takes the whole image as input and with few convolutions creates it's smaller representation (context). This context vector

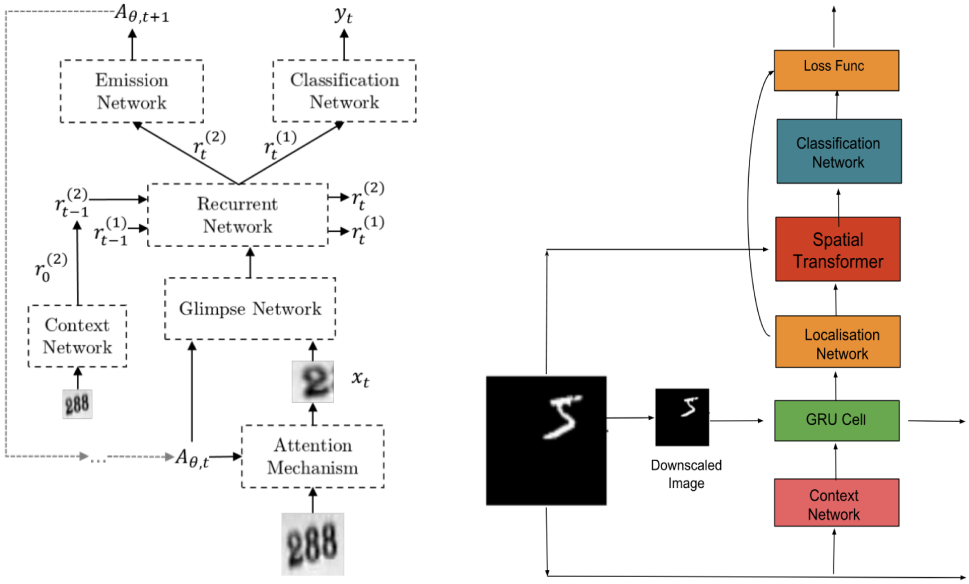


Fig. 2.10: Network architecture of EDRAM [34] and recurrent-STN [35]

is then used by RNN to predict the locations to attend at every iteration. Classification part of the network only sees attended regions.

Both the Network suggest that an attention-based model may be better than an only CNN based model at both dealing with clutter and scaling up to large input images.

Methodology

3

> *Innovation distinguishes between a leader and a follower.*
>
> — **Steve Jobs**
> (CEO Apple Inc.)

This chapter presents the methodological approach employed, architecture of Attentional PointNet (Attention in 3D), the loss function used for training, and an efficient algorithm to implement the network. The chapter assumes prior knowledge of Visual Attention Mechanism [36, 34, 32, 33], Spatial Transformer Network [26], and Recurrent Neural Network [37] and Chapter 2. It is organized as follows: section 3.1 introduces the central idea upon which the proposed Attentional PointNet, outlined in section 3.2, is based.

3.1 Approach

3D pointcloud can be perceived as similar to MNIST cluttered dataset. As most of the space in pointcloud is empty (black in MNIST) and as described in section 4.1.4 each cropped region of point cloud has 0-3 objects of interest (handwritten digits in MNIST) along with clutter.

As in [30] we take inspiration from the human perception system to perform visual sequence recognition task by moving focus from one relevant object to another in the scene. Using the concepts of visual attention mechanism studied in section 2.4 and testing them on MNIST cluttered dataset in section 5.2, we propose to use the same in 3D space applied to point clouds. Attention Mechanism has proved to be very effective in many applications including human activity recognition in videos (spatio-temporal 3D data) as shown in glimpse cloud [38] or digit recognition in cluttered MNIST dataset [35]. It is evident to go forward with attention mechanism.

The basic idea is simple, rather than processing whole point cloud, we propose an approach to sequentially focuses on smaller, relevant regions in 3D space for classification and localization of objects. This cuts down the computation time drastically.

3.2 Proposed Architecture

The proposed architecture of Attentional PointNet as shown in Figure 3.1 consist of four core functional blocks: Data Augmentation block section 3.3, Context Network section 3.4, Recurrent 3D Spatial Transformer Network section 3.5 and 3D object Classifier section 3.6. The network uses raw 3D point clouds generated from high definition LiDARs and predicts bounding boxes for detection of obstacles, such as cars for example. A special loss function was designed for the network to be end to end trainable is elaborated in section 3.7.

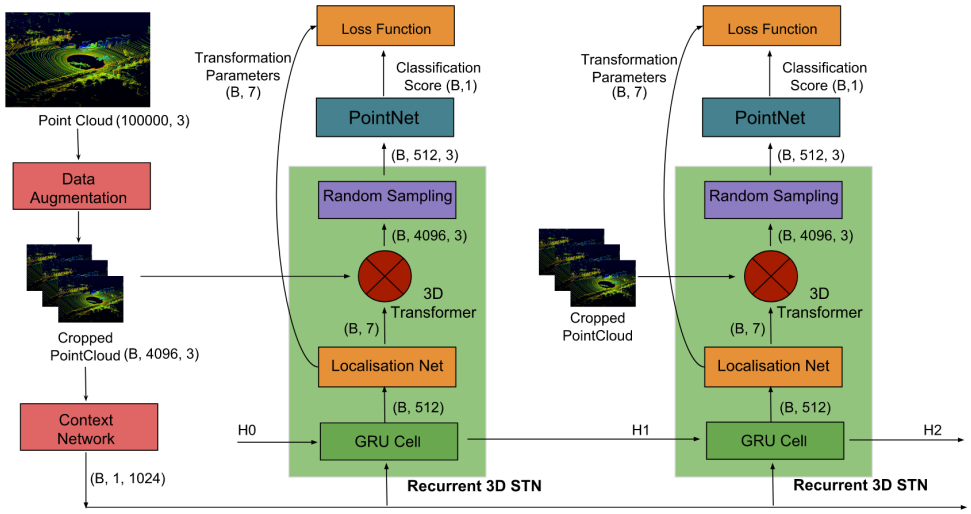


Fig. 3.1: Network architecture of Attentional PointNet

3.3 Data Augmentation Block

Data Augmentation block takes raw pointcloud as input typically consisting of 100k points in R^3. It returns 9 cropped regions of size 10m x 10m. Points in each cropped region are quantified into voxels and re-sampled to contain a fixed 4096 points. Similar strategy as described in section 4.1.4 is used for re-sampling. Only 9 cropped regions are chosen, allowing the network to operate in real-time. Locations of these 9 cropped regions are shown in Figure 4.4.

3.4 Context Network

Context Network is a network which provides contextual understanding of the input point clouds, pointing towards possible objects of interest. Context network using a series of 1D convolution layer (Multi-Layer Perceptron) converts the point set into higher dimensional feature space. A final max pooling layer is used as a symmetric function to aggregate information from all the points. The architecture and training of our Context Network is similar to the T-Net in [18] with some modifications as shown in figure:

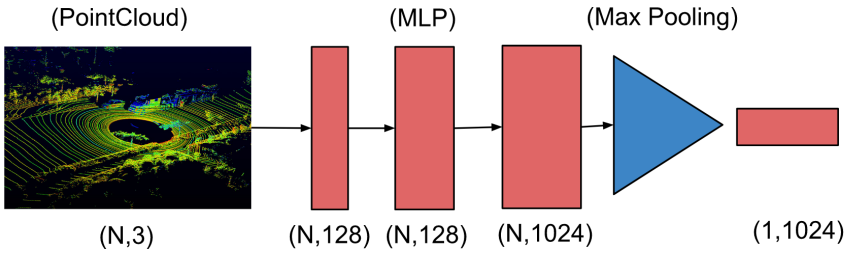


Fig. 3.2: Architecture of Context Network; MLP -Multi-Layer Perceptron

Our Context Network takes 4096 points as inputs and outputs one single vector of size 1024. We discovered that Batch Normalization in our case tends to worsen the results. To increase the output accuracy(at the expense of training speed), the batch normalization layers of the first and last blocks are removed.

As our network directly works on point set, in contrast to pixel arrays in images or voxel arrays in volumetric grids, point sets are unordered. Which requires network taking N number of 3D points as input, needs to be invariant to N! permutations. Furthermore, simple transformations like rotation and transformation should not modify the higher level contextual understanding of point cloud by the network.

To be invariant to permutation and simple transformations (as done in PointNet [18]), we use a using Max Pooling layer as a symmetric function applied to the transformed elements in the set. This render the order of input of point cloud to be insignificant.

$$f(\{x_1, x_2, ..., x_n\}) \approx g(h(x_1), h(x_2), ..., h(x_n)) \tag{3.1}$$

where $f : 2^{\mathbb{R}^N} \rightarrow \mathbb{R}$, $h : \mathbb{R}^N \rightarrow \mathbb{R}^K$ and $g : \mathbb{R}^K \times ... \times \mathbb{R}^K \rightarrow \mathbb{R}$ is a symmetric function.

3.5 Recurrent 3D Spatial Transformer Network

3D recurrent STN is at the heart of proposed Attentional PointNet. It outputs 3D region proposals. This differs from region proposal networks, in a way that the proposals are sequentially generated for different object at each iteration step.

A 3D recurrent STN comprises of a Gated Recurrent Unit(GRU) cell, localisation network, 3D transformer and random sampler. Following sections explain functioning of each block in detail:

3.5.1 Gated Recurrent Unit (GRU)

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks (RNN). GRUs were preferred over vanilla RNNs to avoid the vanishing gradient problems with the latter.

A GRU cell takes input I of size (B,1024) from the context network and h_{t-1} a hidden Tensor from the GRU cell in previous iteration $(t-1)$. It outputs a tensor h_t of shape (B,512) which is used as the input to GRU cell in next Iteration and as the input to localization network.

GRU layer is trained to sequentially produce the location of new object at every iteration step (t). The current hidden state h_t encodes the information about the location of the current and previous objects. This information is passed down to GRU cell at next iteration step as h_{t-1}. Values coming in from I and h_{t-1} are then used to determine the new current hidden state h_t, this enables network to focus on object not seen in the previous iteration steps.

3.5.2 Localization Network

Localization network is a 3 layered fully connected perceptron network. At every iteration it takes an input h_t from GRU cell and regress 7 variables $(\theta_{11}, \theta_{12}, \theta_{21}, \theta_{22}, \theta_{14}, \theta_{24}, \theta_{34}) \in \Theta_t$ of a 3D transformation matrix, of which 4 parameters for rotation $(\theta_{11}, \theta_{12}, \theta_{21}, \theta_{22})$ whereas $(\theta_{14}, \theta_{24}, \theta_{34})$ are for translation along 3 axes.

The rotation matrix along z-axis and transformation matrix can be written as:

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad T(\Theta_t) = \begin{bmatrix} \theta_{11} & \theta_{12} & 0 & \theta_{14} \\ \theta_{21} & \theta_{22} & 0 & \theta_{24} \\ 0 & 0 & 1 & \theta_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

where γ is angle of rotation along z-axis. $\gamma = tan^{-1}(\theta_{21}/\theta_{11})$. We only consider a subset of rigid 3D transformations since rotation about X and Y axes is neglected along with the effects of scale and shear. This is in contrast with images, as the scale/ size of the object does not change with respect to the distance of the object from the sensor. Future work will include the scale parameter to handle the cases when objects belonging to same class are of varying sizes. For simplicity, we have only considered the orientation with respect to z-axis. Unlike the original STN [26] that has no direct supervision on transformation $T(\Theta_t)$, we explicitly supervise our localization network to predict object locations.

This is mathematically formulated as:

$$C = f_{context}(I) \tag{3.3}$$

$$h_t = f_{RNN}(c, h_{t-1}) \tag{3.4}$$

$$T(\Theta_t) = f_{loc}(h_t) \tag{3.5}$$

where $f_{context}$ is the context network taking I as input and outputs a context vector C, f_{RNN} is a GRU cell, and f_{loc} is localization Network. Here, a rigid transformation $T(\Theta_t)$ is produced at each time-step from the hidden state of the RNN. More importantly, the rigid transformations are conditioned on the previous transformations through the time dependency of the RNN.

3.5.3 3D Transformer

3D Transformer is one of the key contribution of this thesis. 3D transformer can be seen as differentiable cropping mechanism for Hard Attention in 3D space. Let the input point cloud in bounding box of size (W, L, H) centered at $(0, 0, 0)$ in $\mathbb{R}^3$ space be transformed such that the points belonging to object of interest lie inside a smaller bounding box of size (W', L', H') centered at $(0, 0, 0)$ in $\mathbb{R}^3$ space as illustrated in Figure 3.3:

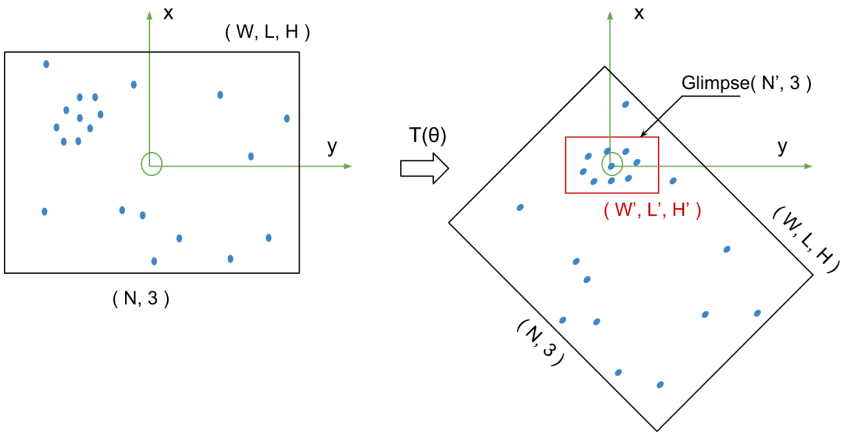


Fig. 3.3: 2D illustration of working of 3D Transformer

As the network attends to the points falling inside the smaller bounding box, it can be called as a 3D glimpse. The layers ahead only use the points contained within this 3D glimpse for processing/classification. More information is detailed in section 5.3.1

More formally, a 3D transformer takes transformation matrix parameters as input from Localization Network and transforms the input pointcloud $P(4096, 3) \rightarrow P'(4096, 3)$. The pointwise rigid 3D transformation is given by:

$$\begin{bmatrix} x_i^t \\ y_i^t \\ z_i^t \\ 1 \end{bmatrix} = T(\Theta_t) \begin{bmatrix} x_i^s \\ y_i^s \\ z_i^s \\ 1 \end{bmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & 0 & \theta_{14} \\ \theta_{21} & \theta_{22} & 0 & \theta_{24} \\ 0 & 0 & 1 & \theta_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^s \\ y_i^s \\ z_i^s \\ 1 \end{bmatrix} \tag{3.6}$$

where (x_i^t, y_i^t, z_i^t) are the transformed coordinates of output pointcloud P', $(x_i^s; y_i^s, z_i^s)$ are the source coordinates of the input pointcloud P, and T_Θ is the rigid transformation matrix.

3.5.4 Random Sampler

A random sampler extracts the points inside the 3D glimpse while discarding all the points outside glimpse. Only a small subset of original point cloud is passed further in the network for the classification. Aiding the network in real-time performance by reducing the data needed to be processed.

The extraction of points in this manner is not a completely differentiable operation as there are no gradients with respect to points outside glimpse, but changing them slightly does not affect the output. This operation can be seen as similar to Max Pooling where the gradient from the next layer is passed back to only neuron which achieved the maximum, whereas all other other neurons get zero gradients.

Furthermore, points inside glimpse are re-sampled to 512 points in a similar strategy as described in section 4.1.4.

3.6 3D Classifier

3D Classifier is another key component of the network essential to categorize the points attended by the network. This module can be replaced by any classifier that directly works on 3D points [18, 20, 12, 39, 40]. In this thesis, we compared PointNet and pointwise convolution Network as described in section 5.1. We opt to use PointNet as our 3D classifier considering the trade offs between accuracy and inference time. The simplicity of PointNet architecture is another compelling reason for this choice.

3.7 Loss Function

For multiple object detection, the network should locate the objects in a point cloud and successfully categorize them. Hence, the loss function should penalize any false positives predictions along with any incorrect recognitions at true positive locations.

For the network to be end to end trainable, a reformulated loss function similar to [34] is proposed. It is an ensemble of two losses: ("loss what")

binary cross-entropy loss for the given glimpse $\mathcal{L}^y$ and ("Loss where") mean squared error of transformation matrix parameters $\mathcal{L}^{T(\Theta)}$.

$$\mathcal{L}_t^y = -(y_{gt} log(p_{t,y_{gt}}) + (1 - y_{gt}) log(1 - p_{t,y_{gt}})) \tag{3.7}$$

$$\mathcal{L}_t^{T(\Theta)} = \delta * \sum_{k=1}^{7} (\theta_{k,t} - \theta_{k,t}^{gt}) \tag{3.8}$$

where $y_{gt} \in \{0, 1\}$ is binary ground truth class, $p_{t,y_{gt}}$ is a predicted class probability on a ground truth position, $(\theta_{1,t}...\theta_{7,t})$ are elements of predicted transformation matrix $T(\Theta_t)$ and $(\theta_{1,t}^{gt}...\theta_{7,t}^{gt})$ are elements of ground truth transformation matrix for RNN iteration step t.

For binary classification of car detection, classification network has to be trained on both positive and negative samples, although the localization network should only learn to locate positive samples. To prevent localization network learning to predict location of false samples, parameter $\delta = \{1, 0\}$ in plays a key role.

$$\delta = \begin{cases} 0 & y_{gt} = 0 \\ 1 & y_{gt} = 1 \end{cases} \tag{3.9}$$

In our augmented training datasetsection 4.1.4, each crop of the point cloud is labelled with 3 locations. According to the number of cars in the crop, the locations could be of a car or a random location (if number of cars less than 3). The locations are also associated with the corresponding class label.

We let our network run for 3 iterations, making 3 predictions. We assign ground truth to predictions using Hungarian algorithm. We create a (3 x 3) cost matrix based on the Euclidean distance between ground truth positions and predicted positions. Assignments between them are found such that the total cost of assignments is minimum. Losses of each iteration are calculated and final loss is average of losses at each iteration.

$$\mathcal{L} = \frac{1}{3} \sum_{t=1}^{3} \alpha * \mathcal{L}_t^{T(\Theta)} + \beta * \mathcal{L}_t^y \tag{3.10}$$

where $\mathcal{L}^{T(\Theta)}$ can be interpreted as “where to look” and $\mathcal{L}^y$ as “what to look”. The hyperparameters α and β allows a trade-off between transformation and

classification loss, forcing the model to simultaneously optimize between a better patch extraction and better recognition.

Data, Software, Hardware

4

> *Not everything that can be counted counts, and not everything that counts can be counted.*
>
> — **Albert Einstein**
> (Theoretical physicist)

4.1 Dataset

In this section we describe about the datasets and data augmentation processes used for the experimentations in this thesis.

4.1.1 Cluttered MNIST Dataset

We used Cluttered MNIST dataset Figure 4.2 from DeepMind for evaluating differentiable Visual Attention Mechanism described in section 2.4.2. The original MNIST Hand written digit dataset is augmented with additional noise and distortion. Every image contains only one digit. The augmentation process involve three steps:

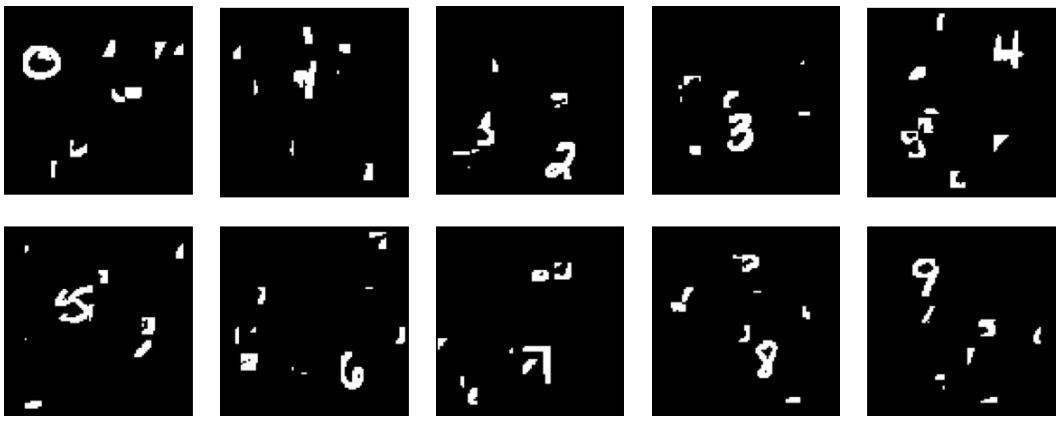


Fig. 4.1: Sample images of cluttered MNIST dataset

1. Randomly select a position and orientation of the original 28x28 image in a 100x100 canvas.

2. Randomly crop 9x9 patch 8 times from a randomly selected 28 x 28 image from training/validation set. Then stitch all the 8 cropped images to augment 100 x 100 image with distortion and noise.

4.1.2 ModelNet40 Dataset

In this thesis we use princeton's ModelNet40 dataset [41] for evaluation of 3D classifiers (PointNet and PointWise Convolution). The dataset contains CAD models of 40 most commonly found object categories. Every object has corresponding point cloud consisting of 2048 points. Objects are centered at origin but are rotated in random orientation. The total dataset is divide as 9840 object point clouds for training and 2468 object point clouds for testing.

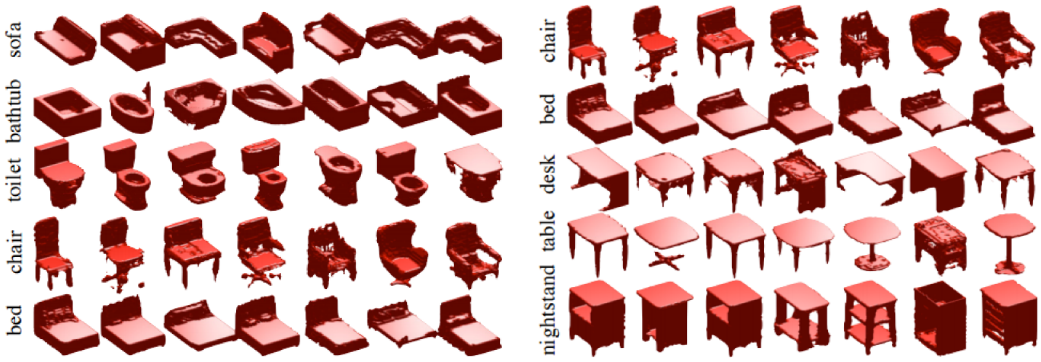


Fig. 4.2: Sample images of ModelNet40 [41] dataset.

4.1.3 ACFR Sydney Urban Dataset

We also used ACFR Sydney Urban Dataset for evaluation of 3D classifiers. This dataset contains common urban road objects scanned with a Velodyne HDL-64E LiDAR, collected in Sydney, Australia. This dataset is comparatively smaller consisting of only 10 categories and 631 individual scans of objects across classes of vehicles, pedestrians, signs and trees. Interesting property of the dataset is that it provides non-ideal sensing conditions that are representative of practical urban sensing systems. Objects are with a large variability in viewpoint and occlusion and density of points.

To be able to use this dataset with our network, We first balance the dataset by weighted sampling of the object categories. Also, to account for variability in number of points in each scan of objects were randomly resampled them with replacement to constant 2048 points.

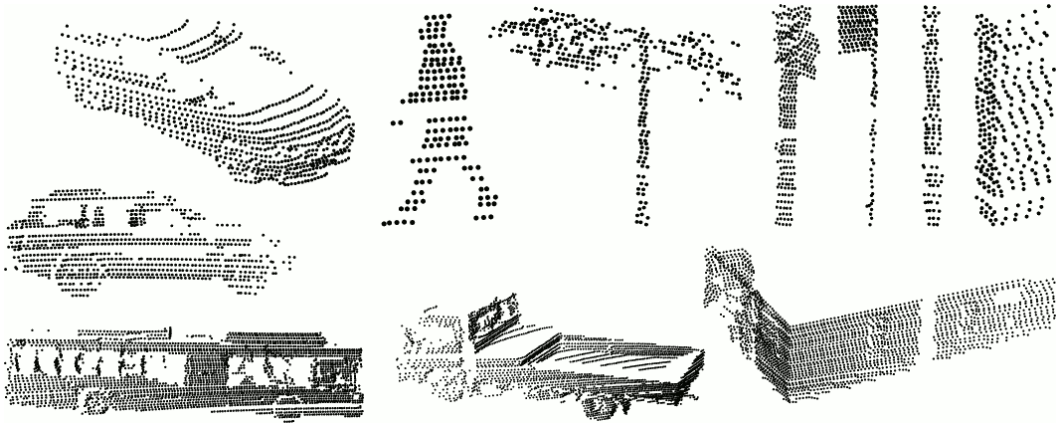


Fig. 4.3: ACFR Sydney Urban Dataset

4.1.4 KITTI 3D object detection Dataset

We use KITTI 3D object detection dataset[42] for evaluation of the proposed Attentional PointNet architecture in this thesis. The KITTI dataset consists of 7481 training images and 7518 test images as well as the corresponding point clouds from Velodyne HDL-64E LiDAR, comprising a total of 80,256 labelled objects. For this thesis work, we only evaluate for 'Car' detection benchmark, as only for cars enough instances for a comprehensive evaluation have been labelled. Each labelled object is associated with 3D object location (x,y,z), 3D object dimensions (H,W,L) and orientation of object θ.

Augmentation of KITTI Dataset: For training of Attentional PointNet a custom dataset was generated by augmenting KITTI dataset. A typical point cloud from High definition LiDAR is composed of $\approx 100k$ points and has range of about 120m. The point density drastically reduces and data becomes unreliable as we go farther from the sensor. In this thesis we work with high point density and feature rich $30m \times 30m$ area in front of car as shown in Figure 4.4. This is to save computation and demonstrate real-time performance of the Network.

To generate our dataset we subdivide the working area form each scan into equally spaced cropped regions of size 10m x 10m as shown in Figure 4.4. We demonstrate working of our 3D Attention Mechanism on these cropped regions.

Each cropped region consists, number of points ranging between 20,000 to none. Directly processing all the points imposes increase in memory/ efficiency burdens on the computing platform, and also highly variable point

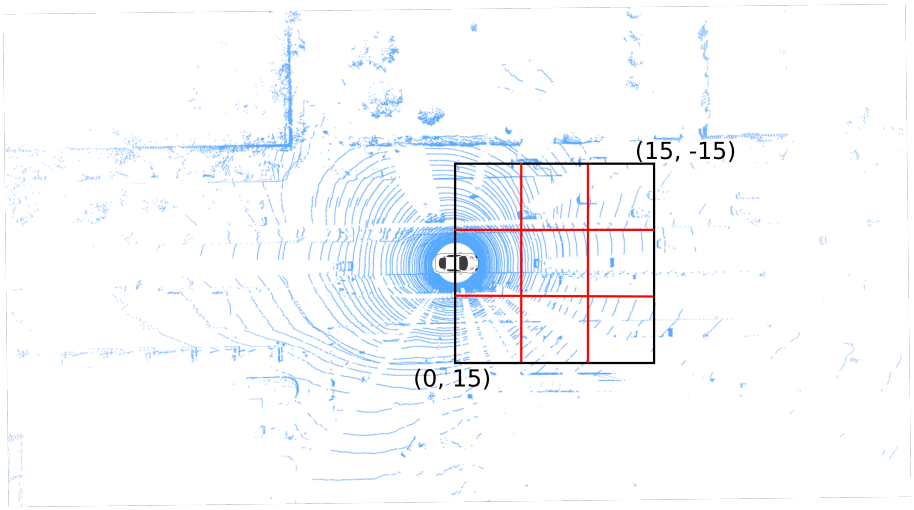


Fig. 4.4: 30m x 30m Working area in blue rectangle, 10m x 10m crops in red squares

density throughout the space might bias the detection. We randomly sample each cropped region to have fixed number, N = 4096, points. The re-sampling strategy used is as described below.

1. To reduce the sampling bias we first voxelize the pointcloud keeping the leaf size to 0.05m.

2. After voxelization step, the cropped regions having points more than 'N' are randomly sampled without replacement to 'N' number of points.

3. The cropped regions with less than 'N' number of points are first multiplied with a factor $'S'$ and then they are randomly sampled without replacement to 'N' number of points. Where,

$$S = \frac{N}{\text{num of points in crop}} + 1$$

4. We discard the cropped regions with number of points less than a certain threshold.

5. Finally, we recentre each cropped region with origin.

The next step involves generating the labels. Each cropped region is labelled with 3 locations (in the form of 7 parameters of transformation $T(\Theta)$) and corresponding class labels for the object at that location (see Figure 4.5). For this thesis we assume that there could be maximum 3 cars in a cropped

region of 10m x 10m area. We use information from KITTI dataset to check if there are any cars inside each cropped regions and we note the position and orientation of them (green bounding box). We discard the crops with more number of cars. If there are less than 3 cars we generate bounding boxes at random locations and mark them as non-car (red bounding box).

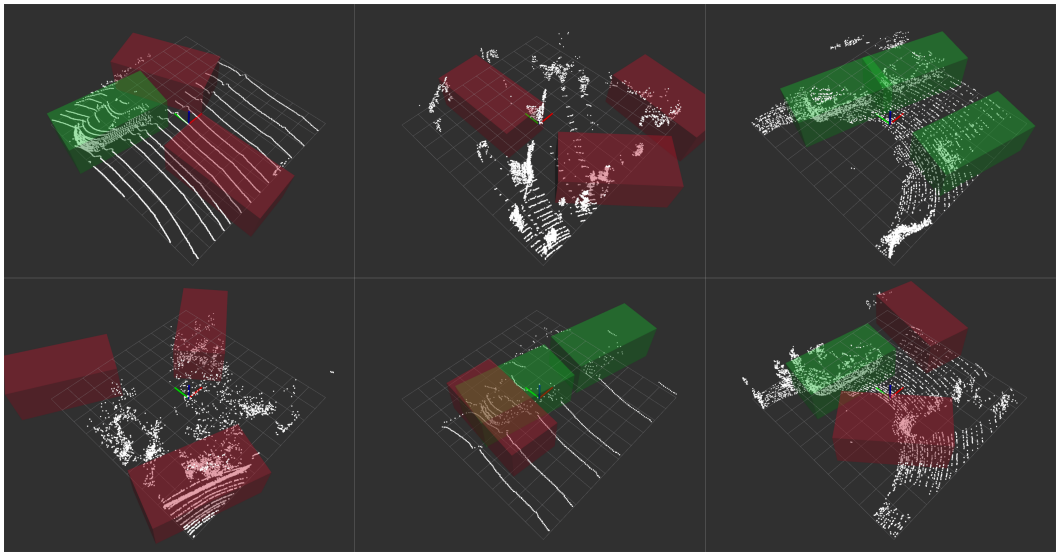


Fig. 4.5: Augmented KITTI dataset, each cropped region has 3 bounding boxes. The green bounding boxes represent the locations of car and red bounding boxes are random locations of non-car regions.

Our augmented KITTI dataset consist of 26,000 crops of 10m x 10m area each having number of cars in a range of 3 to none. We divide the whole dataset as 70/30 percent for training and validation respectively.

4.1.5 3D Pointcloud Dataset(Chroma-Inria)

For the visualization purpose and to demonstrate robustness of the network we use point cloud data from Team Chroma (Inria). The data was acquired using their autonomous vehicle platform, Renault Zoe car Figure 4.6, which has been equipped with a Velodyne HDL-64 on the top, covering a 360-degree field of view (FOV). The dataset contains various scenarios including highways, city-center and busy cross-ways.

Fig. 4.6: Autonomous vehicle platform, Renault Zoe car, equipped with a Velodyne HDL-64 LiDAR

4.2 Software

The entire work in this thesis including proposed methodology, baseline Network architectures, data augmentation codes were implemented in *python* using *PyTorch* deep learning framework. Other libraries including *Numpy* and *Scipy* for numerical processes, *matplotlib* for plotting, *python-PCL* from StrawLab for pointcloud manipulation were also used.

The whole software was designed to be compatible with the Robot Operating System (ROS). As ROS framework act as a bridge between autonomous vehicle hardware and software in Chroma team. In the context of this work, ROS was used to acquire data from LiDAR and connect it with the detection algorithm. Also, ROS-Rviz was used for visualization of 3D pointcloud and the bounding boxes.

4.3 Hardware

For training, validation and testing of proposed Attentional PointNet architecture and other baseline architecture following configuration of the machine was used:

- **CPU** Intel® Xeon(R) CPU W3520 @ 2.67GHz × 4

- **GPU** GeForce GTX 1080 - 8GB
- **Memory** 8GB
- **OS** Ubuntu 16.04 Xenial LTS
- **CUDA version** 9.0
- **PyTorch version** 0.4.0

Experiments and Results

5

> *A theory is merely a scientific idea controlled by experiment.*
>
> — **Claude Bernard**
> French physiologist, historian

The previous chapter 2 describes related works and chapter 3 describes the proposed methodology. The networks detailed in those chapters have multiple parameters and configurations. In this chapter we describe about the network configurations and experiment with parameters and state the results based on different evaluation criteria.

We perform three experiments: section 5.1 describes about experiments with 3D classifiers, section 5.2 describes experiments with differentiable attention mechanism and section 5.3 describes experiments with proposed Attentional PointNet architecture.

5.1 3D Classifier

In this section we evaluate and compare two architectures PointNet [18] and PointWise Convolution Network [20]. As explained in the section 2.2 both the architecture directly consume pointcloud and classify 3D objects. For comparison, we use the implementation of PointNet by Qi *et al.* in PyTorch, whereas for Pointwise Convolution Network we implemented the architecture in PyTorch for the fair comparison. In these experiments, no attention mechanism was used and no localization was performed.

For the evaluation, we used ModelNet4O and ACFR Sydney Urban dataset. Every object in ModelNet40 dataset consists of 2048 points whereas in ACFR dataset, for each object, 512 points were sampled from the object with replacement. While ModelNet40 dataset object point clouds are clean and uniform, the point clouds in the ACFR dataset are more realistic with occlusions and missing data points. Using both the datasets we evaluate the effectiveness and robustness of the architectures.

Dataset: ModelNet40

Model	Data points	Inference Time	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Epochs
PointNet	2048	12.5ms	96.65	87.83	0.066	0.12	30
PointWise Conv Input Attribute = occupancy	2048	31.25ms	100	72.86	0.0038	1.26	90
PointWise Conv Input Attribute = x, y, z	2048	32.5ms	100	52.68	0.006	1.58	90
2 PointWise Conv layer + PointNet	2048	18.75ms	87.4	84.3	0.23	0.56	90

Dataset: ACFR Sydney

Model	Dataset	Inference Time	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Epochs
PointNet	512	7.2ms	94.52	82.46	0.086	0.86	30
PointWise Conv	512	12.5ms	100	78.23	0.063	0.75	90

Tab. 5.1: Evaluation of PointNet and Pointwise convolution network architecture

As inferred from the Table 5.1 the PointNet architecture has overall superior performance in terms of both accuracy and computational time. Moreover, PointWise convolution requires searching the nearest neighbours and sorting them into subdomains which has the computational complexity of O(dN) where N is the number of points and d is the dimensions. As point clouds from high-definition LiDARS have a significantly larger number of points and a real-time implementation is crucial. We decided to use PointNet as our 3D classifier for proposed Attentional PointNet architecture.

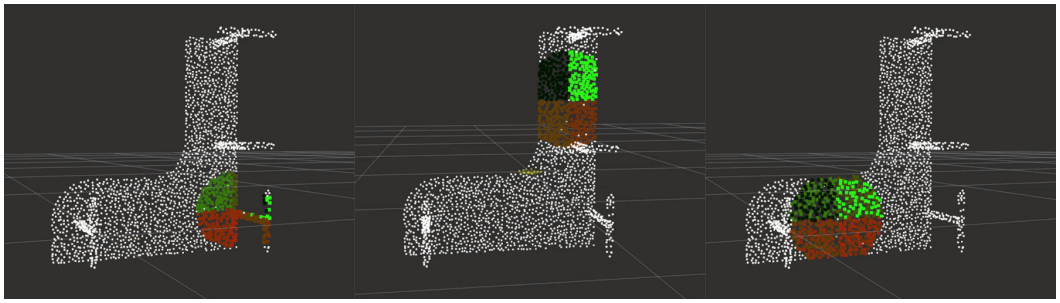
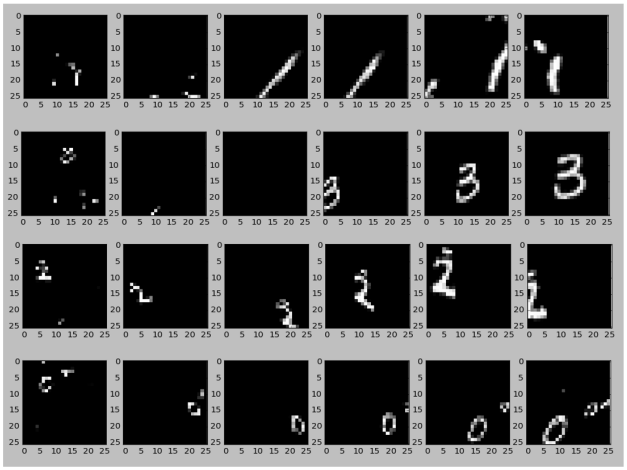


Fig. 5.1: Working of implemented point-wise convolution operator on point cloud of a table from ModelNet40 dataset. Shows partitioning of point cloud by PointWise kernel into subdomains represented by different color. Points within a subdomain share same weight.

5.2 Visual Attention Mechanism

In this section, we evaluate and compare two architectures of differentiable visual attention Mechanism EDRAM [34] and Recurrent-STN [35]. We implemented both the architectures in PyTorch.

To evaluate the effectiveness of the architectures we first investigate a single object detection tasks involving MNIST cluttered dataset. In this dataset, we have 44800 training and 12600 testing images with a single handwritten digit randomly placed along with the clutter. The networks have to learn to focus on the digits (avoiding the clutter) and then classify the digit accurately.

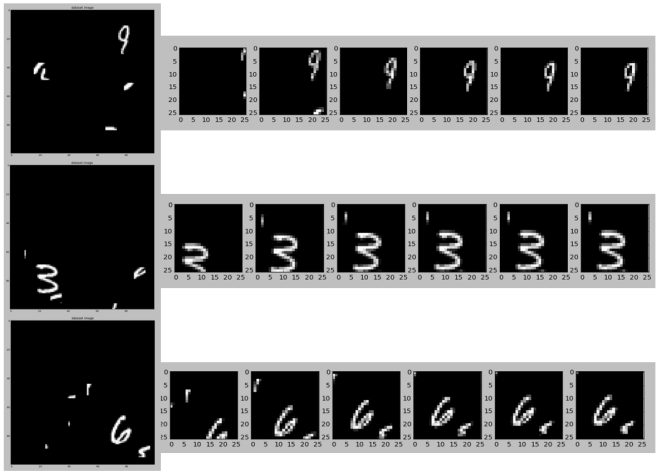


Dataset	Mnist Cluttered
Samples	44800/12600
Train Accuracy	98.45
Test Accuracy	85.67
Train Loss	0.86
Test Loss	1.63
Epochs	90
Loss Func	Loc + Class

Fig. 5.2: EDRAM Network: Each row shows a sequence of glimpses taken by the network while recognizing MNIST Cluttered dataset.

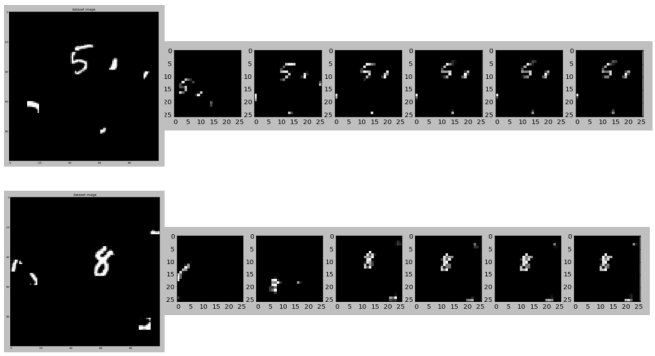
Figure 5.2 illustrates working of EDRAM network. The original input image is of 100 x 100 pixels while each glimpse is of 25 x 25 pixels. Here each row shows how the Spatial Transformer based attention mechanism locates a digit on the image across iterations. In the first glimpse, the network sees the whole image then zooms in on the digit at every iteration. Results in Figure 5.2 shows high training accuracy but low testing accuracy. On training further, we found that though the classification accuracy is high the network is not good in localizing the digit itself. One possible explanation of this is as MNIST dataset is simpler the network could recognise the digit from the first glimpse itself and do not learn to localize digit.

Figure 5.3 and Figure 5.4 shows the working of recurrent-STN [35] architecture. To this network, we make minor modifications by adding localization loss to final cross entropy classification loss. We then compare the results of



Dataset	Mnist Cluttered
Samples	44800/12600
Train Accuracy	98.43
Test Accuracy	93.75
Train Loss	0.124
Test Loss	0.263
Epochs	60
Loss Func	Loc + Class

Fig. 5.3: Recurrent-STN Network with supervised localization



Dataset	Mnist Cluttered
Samples	44800/12600
Train Accuracy	93.75
Test Accuracy	88.75
Train Loss	0.523
Test Loss	0.505
Epochs	60
Loss Func	Class

Fig. 5.4: Recurrent-STN Network without supervised localization

the modified network (fully supervised localization) and original network (unsupervised localization). The tables in Figure 5.3 and Figure 5.4 demonstrate that with full supervision over localization the network accuracy is higher and network tends to localize the digits well. In the other case, rather than zooming on to the digits, it zooms out to sees the whole picture and learns the class from it. This concludes that the supervision over localization, network is essential.

5.3 Attentional PointNet

In this section, we explain the implementation details of proposed Attentional PointNet and the training procedure.

5.3.1 Network Details

For car detection task, we consider point clouds within a range of $[0, 30] \times [-15, 15] \times [-2, 1]$ meters along X, Y, Z axis respectively with X-axis is aligned with the orientation of the car. We then divide this region into equally spaced cropped regions of 10m x 10m as explained in section 4.1.4. Points in each cropped region are randomly sampled to 4096 points and used as input for Attentional PointNet.

The context network consists of three fully connected layers implemented as 1D convolutions with input-output feature sizes as (3,64),(64,128) and (128,1024) respectively for each layer. For each input point, we only use x,y,z co-ordinate values as attributes. a ReLU layer and a batch normalization layer, apart from the first layer. Finally, a Max Pooling layer aggregates features from all 4096 points into one vector of feature size 1024. This vector act as the context of input point cloud. At every iteration, GRU cell uses this context vector as input. For the first iteration, input hidden state vector is initialized with zeroes and GRU cell returns a vector of size 512.

The localization network uses this vector to regress 7 transformation parameters $\{\theta_{11}, \theta_{12}, \theta_{21}, \theta_{22}, \theta_{41}, \theta_{42}, \theta_{43}\} \in T(\Theta_i)$. Localization Network also consists of three fully connected layers with (input, output) size pairs as (512,256), (256,128), (128,7) respectively. Only first two layers include ReLU whereas only the first later includes batch normalization. Transformation matrix is first initialised as an 4x4 identity matrix.

The 3D transformer uses these parameters to translate and rotate 4096 input points. The random sampler then extracts all the points inside the sampling box of size 2.5m x 5m x 2m centered at the origin as described in section 3.5.4. This size was chosen to approximately represent the size of average cars. Points inside the sampling box are randomly sampled to 512 points.

We use a cut downed version of PointNet [18] as a 3D classifier. We remove the T-net and alignment network from PointNet architecture and use only classification part. This is because our recurrent 3D STN module (as explained in section 3.5.3) already performs the task of aligning object point cloud. The final layer of PointNet is modified to output binary classification.

In the loss function values of hyperparameters α and β are set to 0.7 and 0.3 respectively giving more weightage to finding correct location of the object.

For training of the network, we use stochastic gradient descent (SGD) algorithm with momentum of 0.9 and batch size of 32. We keep learning rate to 0.01 for first 30 epochs and then lower it to 0.001 for further epochs. Training on our custom KITTI dataset takes 8 to 9 hours to converge (250 epochs) with PyTorch and GTX 1080GPU.

5.3.2 Training of Network

For the training of Attentional PointNet we use our augmented KITTI dataset as described in section section 4.1.4. For training, we have 18,200 point cloud samples of cropped regions of size 10m x 10m. Each sample point cloud consist of 4096 points and each point is associated with its x, y, z coordinates. Each sample point cloud is labelled with 3 locations (in a form of 7 parameters for transformation $T(\Theta_t)$). According to the number of cars in the sample point cloud, the locations could be of a car or a random location (if number of cars is less than 3). The locations are also associated with the corresponding class label, 1 for car and 0 for random location.

With the batch size of 32, input to the network are the point cloud data of size (32,4096,3) and corresponding labels of size (32,3,8). We allow network to iterate 3 times, predicting location of new car at each iteration step. If there are less than 3 cars in the sample point cloud, the network outputs a random location in the remaining iterations. The network also outputs class probability of object at predicted location. Final output of the network is of the size (32,3,8).

In the loss function for each sample point cloud, ground truth and detections are matched using Hungarian algorithm. We create a (3 x 3) cost matrix based on the Euclidean distance between ground truth positions and predicted positions. Assignments between them are found such that the total cost of assignments is minimum.

For validation, we have 7800 sample point clouds and we optimise our hyper-parameters on this subset of dataset.

5.3.3 Evaluation on KITTI

We evaluate Attentional PointNet on the KITTI 3D object detection benchmark [42] which contains 7,481 training images/pointclouds and 7,518 test images/point clouds. In this thesis work we only evaluate for Car category. As KITTI dataset do not provide the ground truth for the test set and the access to their test server is limited, we evaluate Attentional PointNet using the protocol described in [17, 16].

Evaluation Metrics: We assess the performance of our network as per metrics defined in [42]. For jointly detecting objects and estimating their 3D orientation we provide results for average orientation similarity (AOS), which is defined as:

$$AOS = \frac{1}{11} \sum_{r \in \{0, 0.1, ..1\}} \max_{\tilde{r}:\tilde{r} \geq r} s(\tilde{r}) \tag{5.1}$$

Here, $r = \frac{TP}{TP+FN}$ is the PASCAL object detection recall, where detected 2D bounding boxes are correct if they overlap by at least 50% with a ground truth bounding box. The orientation similarity $s \in [0, 1]$ at recall r is a normalized $([0..1])$ variant of the cosine similarity defined as

$$s(r) = \frac{1}{|D(r)|} \sum_{i \in D(r)} \frac{1 + cos\Delta_{\theta}^{(i)}}{2} \delta_i \tag{5.2}$$

where $D(r)$ denotes the set of all object detections at recall rate r and $\Delta_{theta}^{(i)}$ is the difference in angle between estimated and ground truth orientation of detection i. To penalize multiple detections which explain a single object, we set $\delta_i = 1$ if detection i has been assigned to a ground truth bounding box (overlaps by at least 50%) and $\delta_i = 0$ if it has not been assigned.

Finally we also provide Average Precision and pure binary classification results for Car category.

5.3.4 Results

3D detection is a more challenging task as it requires finer localization of objects in 3D space. For the Car category, we compare the proposed method with several top-performing algorithms, including image based approaches: Mono3D [16] and 3DOP [43]; LiDAR based approaches: VeloFCN [44],and VoxelNet [24]; and multi-modal approach: MV3D [17] and Frustum PointNet [23]. We train Attentional-PointNet from scratch using only the LiDAR data provided in KITTI.

Dataset: KITTI Car detection Benchmark

Model	Modality	Inference Time	Average Precision
Mono3D	Mono	-	2.53
3DOP	Stereo	-	6.55
VeloFCN	LiDAR	Not real-time	15.20
MV3D (BV + FV)	LiDAR	Not real-time	71.19
MV (BV + FV+ RGB)	LiDAR + Mono	Not real-time	71.29
VoxelNet	LiDAR	225ms	81.97
Frustum PointNet	LiDAR + Mono	-	88.16
Attentional PointNet (ours)	LiDAR	**55ms**	**63.52**

b

Tab. 5.2: Performance comparison in 3D detection: average precision (in %) on KITTI validation set. BV - Birdeye View, FV - Front view

Table 5.2 summarizes the comparison for the Car category, Attentional PointNet significantly outperforms all other approaches in terms of inference time. Our network also achieves comparable Average Precision (AOS) of 63.52% at recall rate of 0.54 on the subset of KITTI dataset we created for the testing.

Figure 5.5 illustrates the working of Attentional PointNet on 10m x10m cropped regions we extracted from KITTI dataset. For each cropped region, the network makes three predictions, sequentially classifying and localizing the cars in the scene. It can be observed that the network is effective and capable of attending/finding multiple Cars even in highly cluttered environment. When there are no cars in the scene network makes random location predictions and appropriately classify them as negative detections.

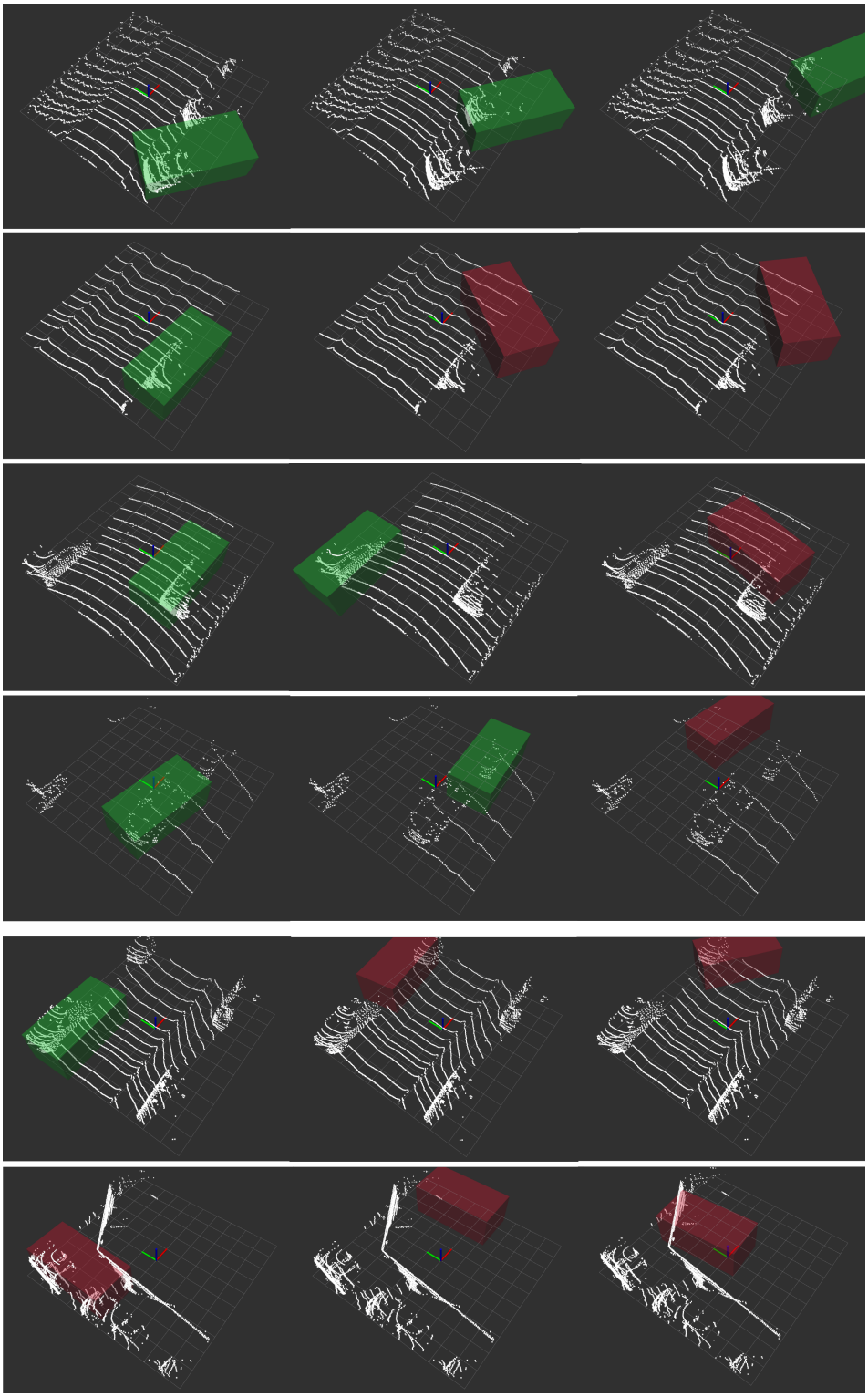


Fig. 5.5: Attentional PointNet: Each row shows a sequence of the predictions by the network. First four rows illustrates scenarios where network successfully identified and localised the car. Whereas last two rows shows failure cases.

Convergence:

Figure 5.6 shows two graphs. First graph show the convergence of the network, training loss reducing over the epochs. While the second graph shows the Precision Vs Recall curve.

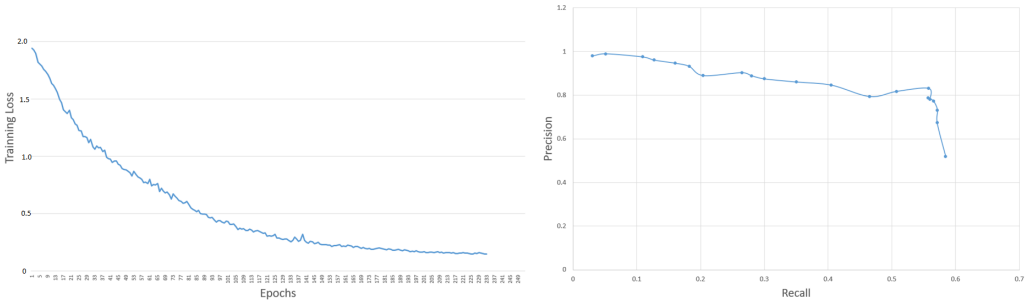


Fig. 5.6: *(a)*Training loss Vs epochs, *(b)*Precision Vs Recall curve.

Inference time:

To process an area of 30m x 30m consisting of ≈ 30K points the inference time is 50ms. Network outputs bounding boxes at 20Hz with GTX 1080 GPU and hardware specified in section 4.3.

5.3.5 Visualisation on Chroma-Inria dataset

For qualitative analysis, checking robustness and practicality of the network, we visualised the network performance on Chroma-Inria dataset. The dataset consist of continuous stream of point clouds at 10 Hz collected with a 64 layered high-definition LiDAR. Figure 5.7 shows network is capable of detecting multiple cars in unstructured environment like highways and crossroads.

On this dataset, we observe many false positive detections by the network mostly in the bushes and off-road areas. Also, when the cars are close to each other, network struggles to distinguish between them and predicting their orientations incorrectly.

Fig. 5.7: Attentional PointNet on Team Chroma Inria dataset

6 Conclusions and recommendations

> *Problems worthy of attack / Prove their worth by hitting back.*
>
> **— Piet Hein**
> (Mathematician, physicist, inventor)

The motivation for this research was the challenge of multiple object detection in 3D pointcloud obtained from high-definition LiDAR on Autonomous Vehicles. Aiming to translate the detected objects into semantic occupancy grids. Real-time performance of perception system being crucial for high-speed Autonomous Vehicles, adds up another challenge to design algorithms efficient to process large data ≈ 100K points at 10 Hz.

In this thesis work, we analysed the properties of point clouds and surveyed the existing methodologies for object classification and localization in 3D point clouds. We implemented PointNet [18] and Point-Wise Convolution Network [20] in PyTorch and evaluated them on ModelNet40 dataset. Upon experimentation, we infer that PointNet shows superior performance than pointwise convolution network in terms for overall classification accuracy and inference time.

The next task was scaling PointNet classifier to detect multiple objects in the cluttered environment by still being real-time. To this end, we explored the techniques to save computational overhead for multiple object detection. Inspiring from how humans perceive a scene, by sequentially focusing only on the relevant regions and creating the understanding of the scene, we studied the use of Visual Attention mechanism in 2D images for multiple object detections. We implemented Spatial Transformer based two differentiable Attention Mechanisms, EDRAM and recurrent-STN in PyTorch and evaluated them on cluttered MNIST dataset. we Inferred that the recurrent-STN performs better at localizing and classifying the hand-written digits in the cluttered MNIST dataset.

As the research contribution of this thesis, we extended the theory of Visual Attention Mechanism to 3-dimensional space.

In the conclusion section 6.1, we summaries our research work and experiments and in recommendations section 6.2 we give insight about possible future work and improvements in the Attentional PointNet architecture.

6.1 Conclusions

Most existing methods for LiDAR based 3D object detection either rely on hand-crafted feature representations like bird's eye view projection or multi-modal approaches using cameras as additional sensor. In this thesis, we present a novel end-to-end trainable deep architecture Attentional-PointNet for 3D object detection in point clouds. The network directly consumes sparse 3D points and capture 3D geometric information effectively. We proposed to use Attention Mechanism with 3D point clouds and introduce a new differentiable module, recurrent 3D-STN, to find regions of interest in the given point cloud. We conducted experiments with augmented KITTI dataset for car detection. We demonstrate our network capability to sequentially attend/locate to cars in the cluttered environments. For car detection on KITTI dataset Attentional PointNet shows comparable results with existing state-of-the-art LiDAR-based 3D detection methods whereas it significantly outperforms them in terms of inference time.

6.2 Recommendations

There are several research avenues can be explored and improvements can be incorporated as:

- Network currently iterates 3 times and predict detection even if there are no objects in the scene. Instead, use of confidence parameter as stopping criteria for the RNN iteration can save computation.

- The original STN paper shows that it helps the network to be inherited / compensated to a small transformation of input data. Thereby making the network less susceptible to changes in viewpoints distortion of input data, increasing the accuracy of the Network. But in our context,

we used STN as a differentiable cropping mechanism for the visual attention. Consequently STN has to predict for the large transformation of the point cloud to focus on different objects. STN suffers at regressing precise bounding box co-ordinate. A better approach could be to increase the crop area of STN (i.e Finding the regions of interest) then designing another network that works inside the cropped out region to regress the bounding box co-ordinate.

- Points in each 10m x 10m cropped regions are randomly sampled to 4096 points, even if there are significantly less number of points in the beginning. This means many points are picked multiple times, this causes a worthless increase in data points to be processed. Design of network that could adapt to the variable number of input points without resampling them would be an interesting challenge to work upon.

- Qualitative results of network show large number of false positives mostly in the bushes. As understanding the shape from such highly unstructured point clouds is difficult, addition of another modality could help reducing number of false positives significantly.

References

[1] Alberto Broggi, Alex Zelinsky, Umit Ozguner, and Christian Laugier. “Handbook of Robotics 2nd edition, Chapter 62 on ”Intelligent Vehicles””. In: *Handbook of Robotics 2nd Edition*. Ed. by Bruno Siciliano and Oussama Khatib. Springer, July 2016, pp. 1627–1656 (cit. on p. 1).

[2] Yong-Joo Oh and Yoshio Watanabe. “Development of small robot for home floor cleaning”. In: *SICE 2002. Proceedings of the 41st SICE Annual Conference*. Vol. 5. IEEE. 2002, pp. 3222–3223 (cit. on p. 1).

[3] Youngmin Park, Vincent Lepetit, and Woontack Woo. “Multiple 3D object tracking for augmented reality”. In: *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society. 2008, pp. 117–120 (cit. on p. 1).

[4] Franz Leberl, Arnold Irschara, Thomas Pock, et al. “Point clouds”. In: *Photogrammetric Engineering & Remote Sensing* 76.10 (2010), pp. 1123–1134 (cit. on p. 1).

[5] Amaury Nègre, Lukas Rummelhard, and Christian Laugier. “Hybrid Sampling Bayesian Occupancy Filter”. In: *IEEE Intelligent Vehicles Symposium (IV)*. Dearborn, United States, June 2014 (cit. on p. 1).

[6] Lukas Rummelhard, Amaury Negre, and Christian Laugier. “Conditional Monte Carlo Dense Occupancy Tracker”. In: *18th IEEE International Conference on Intelligent Transportation Systems*. Las Palmas, Spain, Sept. 2015 (cit. on p. 2).

[7] J. Rios-Martinez, A. Spalanzani, and C. Laugier. “From Proxemics Theory to Socially-Aware Navigation: A Survey”. In: *International Journal of Social Robotics* 7.2 (Apr. 1, 2015), pp. 137–153 (cit. on p. 2).

[8] Özgür Erkent, Christian Wolf, Christian Laugier, David Serra Gonzalez, and Victor Romero Cano. “Semantic Grid Estimation with a Hybrid Bayesian and Deep Neural Network Approach”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2018 (cit. on pp. 2, 4).

[9] Muhammad Zeeshan Zia, Michael Stark, and Konrad Schindler. “Are cars just 3d boxes?-jointly estimating the 3d shape of multiple objects”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3678–3685 (cit. on p. 3).

[10] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. “Complex-YOLO: Real-time 3D Object Detection on Point Clouds”. In: *arXiv preprint arXiv:1803.06199* (2018) (cit. on p. 3).

[11] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. “Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud”. In: *arXiv preprint arXiv:1710.07368* (2017) (cit. on pp. 3, 4).

[12] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. “Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks”. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 1355–1361 (cit. on pp. 3, 25).

[13] Daniel Maturana and Sebastian Scherer. “Voxnet: A 3d convolutional neural network for real-time object recognition”. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 922–928 (cit. on p. 3).

[14] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. “Octnet: Learning deep 3d representations at high resolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 3. 2017 (cit. on p. 3).

[15] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. “O-cnn: Octree-based convolutional neural networks for 3d shape analysis”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 72 (cit. on p. 3).

[16] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, et al. “Monocular 3d object detection for autonomous driving”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2147–2156 (cit. on pp. 4, 41, 42).

[17] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. “Multi-view 3d object detection network for autonomous driving”. In: *IEEE CVPR*. Vol. 1. 2. 2017, p. 3 (cit. on pp. 4, 41, 42).

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1.2 (2017), p. 4 (cit. on pp. 4, 10, 21, 25, 35, 39, 46).

[19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5099–5108 (cit. on p. 4).

[20] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. “Pointwise convolutional neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 984–993 (cit. on pp. 4, 10–12, 25, 35, 46).

[21] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. “PointCNN”. In: *arXiv preprint arXiv:1801.07791* (2018) (cit. on p. 4).

[22] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. “Exploring spatial context for 3d semantic segmentation of point clouds”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 716–724 (cit. on p. 4).

[23] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. “Frustum PointNets for 3D Object Detection From RGB-D Data”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018 (cit. on pp. 4, 42).

[24] Yin Zhou and Oncel Tuzel. “Voxelnet: End-to-end learning for point cloud based 3d object detection”. In: *arXiv preprint arXiv:1711.06396* (2017) (cit. on pp. 4, 42).

[25] Itzik Ben Shabat. *3D Point Cloud Classification using Deep Learning – Recent Works*. http://www.itzikbs.com/3d-point-cloud-classification-using-deep-learning (cit. on pp. 8, 9).

[26] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. “Spatial transformer networks”. In: *Advances in neural information processing systems*. 2015, pp. 2017–2025 (cit. on pp. 10, 15, 16, 19, 23).

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99 (cit. on p. 13).

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788 (cit. on p. 14).

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al. “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37 (cit. on p. 14).

[30] John K Tsotsos. “Analyzing vision at the complexity level”. In: *Behavioral and brain sciences* 13.3 (1990), pp. 423–445 (cit. on pp. 14, 19).

[31] RA Rensink. “The dynamic representation of scenes. Visual Cognition7: 1742. Visual search for change: A probe into the nature of attentional processing”. In: *Visual Cognition* 7 (2000), p. 34576 (cit. on p. 14).

[32] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. “Recurrent models of visual attention”. In: *Advances in neural information processing systems*. 2014, pp. 2204–2212 (cit. on pp. 14, 15, 19).

[33] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. “Multiple object recognition with visual attention”. In: *arXiv preprint arXiv:1412.7755* (2014) (cit. on pp. 14, 17, 19).

[34] Artsiom Ablavatski, Shijian Lu, and Jianfei Cai. “Enriched deep recurrent visual attention model for multiple object recognition”. In: *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE. 2017, pp. 971–978 (cit. on pp. 17–19, 25, 37).

[35] Søren Kaae Sønderby, Casper Kaae Sønderby, Lars Maaløe, and Ole Winther. “Recurrent spatial transformer networks”. In: *arXiv preprint arXiv:1509.05329* (2015) (cit. on pp. 17–19, 37).

[36] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. “Draw: A recurrent neural network for image generation”. In: *arXiv preprint arXiv:1502.04623* (2015) (cit. on p. 19).

[37] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop. 2014 (cit. on p. 19).

[38] Fabien Baradel, Christian Wolf, Julien Mille, and Graham W Taylor. “Glimpse clouds: Human activity recognition from unstructured feature points”. In: *Computer Vision and Pattern Recognition (CVPR)(To appear)* 3 (2018) (cit. on p. 19).

[39] Hang Chu, Wei-Chiu Ma3 Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. “SurfConv: Bridging 3D and 2D Convolution for RGBD Images”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3002–3011 (cit. on p. 25).

[40] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. “Attentional ShapeContextNet for Point Cloud Recognition”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018 (cit. on p. 25).

[41] Zhirong Wu, Shuran Song, Aditya Khosla, et al. “3d shapenets: A deep representation for volumetric shapes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920 (cit. on p. 29).

[42] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012 (cit. on pp. 30, 41).

[43] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, et al. “3d object proposals for accurate object class detection”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 424–432 (cit. on p. 42).

[44] B. Li. “3D fully convolutional network for vehicle detection in point cloud”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017 (cit. on p. 42).

List of Figures

List of Tables